AFRL-IF-RS-TR-2007-24
Final Technical Report
January 2007

AN OPERATIONAL DYNAMIC SITUATIONAL ASSESSMENT CAPABILITY

RAM Laboratories, Inc.

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the Air Force Research Laboratory Rome Research Site Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-IF-RS-TR-2007-24 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
DAWN TREVISANI Work Unit Manager	JAMES W. CUSACK Chief, Information Systems Division Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JAN 2007	Final	Feb 06 – Nov 06

4. TITLE AND SUBTITLE

AN OPERATIONAL DYNAMIC SITUATIONAL ASSESSMENT CAPABILITY

5a. CONTRACT NUMBER
FA8750-06-C-0014

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62702F

6. AUTHOR(S)

Robert McGraw

5d. PROJECT NUMBER
459S

5e. TASK NUMBER
N6

5f. WORK UNIT NUMBER
02

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
RAM Laboratories, Inc.
10525 Vista Sorrento Parkway
San Diego CA 92121-2766

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/IFSB
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2007-24

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 07-024

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The primary objective of the DSAP infrastructure is to allow Commanders and their staff at AOCs the ability to perform 'what-if' analysis of plans and alternatives 'on-the-fly' while continuing augmenting the real-time picture sensor inputs with simulated state-estimated assessments. Enhancements to current DSAP Framework and software infrastructure were made to provide dynamic situational awareness and an improved predictive capability that can be applied to an operational setting. RAM integrated the DSAP Multiple Replication Framework utilizing JSAF with real-time databases and data link simulation provided by Theater Battle Management Core System (TBMCS) and other C41 systems. RAM compared a real-time simulation with a simulation calibrated by real-time-picture inputs from TBMCS and integrated the current predictive analysis capability with the dynamic situational awareness capability for improved calibration of predictive inputs. They implemented interfaces between DSAP components that allow its use in a Service-Oriented Architecture (SOA). This infrastructure will allow commanders to dynamically evaluate and assess the situation in a timely fashion, incorporate prediction in real-time via faster than real-time simulation, support real-time dynamic planning to address targets of opportunity and support a SOA.

15. SUBJECT TERMS
Dynamic Situation Assessment and Prediction (DSAP), software infrastructure, predictive simulation, operationally focused simulation, C2 Decision Support, software frameworks, decision aid technologies

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dawn Trevisani
U	U	U	UL	37	19b. TELEPHONE NUMBER (*Include area code*)

Table of Contents

List of Figures

1.0 Introduction

In order to develop decision science technologies for future Air Operations Centers (AOCs), the Air Force Research Laboratory has defined a program to conduct and sponsor research and development of revolutionary decision-support concepts. A goal of this program is to apply advanced information technologies to promote tactics, techniques and procedures that enable situational awareness and predictive capabilities for future AOCs. One effort that addresses this goal involves the development of a Dynamic Situation Assessment and Prediction (DSAP) Framework. To address predictive analysis of plans in the area of decision support, RAM Laboratories has developed a DSAP Framework and its underlying Multiple Replication Framework (MRF) for AFRL/IFSB. The DSAP Framework leverages the state of the art in advanced information management techniques in the fields of simulation, distributed computing, and information management to provide the underlying functionality to evaluate Commander's plans and their alternatives using predictive simulation while calibrating with the real-time Command Control Communications and Computers Intelligence (C4I) picture. This Final Report details efforts of RAM Laboratories in developing an Operational Situational Awareness Capability based on this DSAP software infrastructure.

1.1 Significance to the Air Force

The DSAP concept grew out of John R. Surdu's Simulation in Operations research project and prototype system (OpSim). There are two basic functions of the DSAP concept: (1) Dynamic Situation Awareness, and (2) Prediction. The overall concept involves the use of embedded simulation in an operational environment to support decision-makers in the planning process. Providing this capability allows decision makers to use simulation to assist in planning operations, monitor current operations, determine deviations from a plan, and predict and determine outcomes of a given plan. The resulting system allows military Commanders to utilize timely battlefield information to make accurate decisions based on the effects of their plans and the current operational picture while providing an underlying capability to allow the Commander to dynamically modify an existing plan "on the fly" to address emerging information detected by sensors or provided by gathered intelligence. Specifically this concept can be leveraged by the Commander's Predictive Environment (CPE) to enable the prediction of likely future events while considering plan options within the context of the mission space, predicted enemy intent, actions, and emerging threats. In addition, through its dynamic situational awareness capability, the DSAP Framework strives to augment sensor information in the operational domain by using real-time simulation to estimate the internal state of resources and assets that may not be visible from current real-time-picture information.

The DSAP Framework builds on advanced information technologies to provide a predictive analysis of existing plans and alternatives within the context of the real-time operational picture. There are five main components of the DSAP Framework: (1) the Multiple Replication Framework (MRF) that is used for evaluating plans and alternative Courses of Action (COAs) against campaign objectives on available processors, (2) the Simulation Framework that allows real-time simulation and faster-than-real-time simulations to be developed in a manner that calibrates with real-time data and supports rollback/rollforward capabilities that can be used to track the real-time operational picture, (3) an Optimization Framework that supports plan generation through the use of simulation-in-the-loop, (4) the simulation component which simulates plans/COAs and their alternatives in both a real-time and predictive fashion (through

the use of faster-than-real-time simulation, and (5) real-time databases and data feeds. This framework provides the capability to dynamically assess situations, dynamically predict the outcomes of plans, and will eventually be used to enhance the plan generation process.

RAM Laboratories, with the Air Force Research Laboratory previously developed a prototype DSAP Framework that provides a predictive capability through the use of faster-than-real-time predictive simulation using JSAF, and real-time data for calibration from the Theater Battle Management Core System (TBMCS) Air Operations Database (AODB) and Modernized Integrated Database (MIDB). This effort built on the previously developed capability to improve dynamic situational awareness by implementing and integrating real-time simulation components with additional real-time databases and data feeds and the existing predictive analysis capability. This effort also installed the DSAP Framework in a laboratory setting to provide a demonstratable capability that can be transitioned to the operational domain and used as a building block for implementing additional decision-support technologies.

The subsequent sections of this Technical Report discuss the further implementation of the DSAP software infrastructure. This work includes implementing the dynamic situational awareness piece of DSAP in support of operations while also further developing the predictive analysis functionality. The remainder of this report discusses the following items:

- Section 2.0 discusses the state of the DSAP Framework at the start of the effort to present the context in which this project was conducted
- Section 3.0 discusses the design, modification, and implementation of a Multiple Replication Framework (MRF) to support dynamic situational awareness support for operations.
- Section 4.0 details work performed on this effort to transition DSAP for use on the Global Information Grid (GIG).
- Section 5.0 outlines future work.
- Section 6.0 provides the Bibliography for this Final Report.
- Section 7.0 defines the acronyms used in developing this report.

2.0 DSAP and the MRF for Predictive Operations

RAM Laboratories has developed the DSAP Framework to address predictive analysis of plans and Courses of Action in order to provide inputs to the operational C2 environment at Air Operations Centers. This effort, builds on the DSAP Framework and its underlying Multiple Replication Framework (MRF) to provide a dynamic situational awareness capability. This section discusses the previously implemented MRF that supports predictive analysis of plans. This section is meant to provide the reader of this Final Technical Report with the basis for the work that was performed under this effort and documented in Sections 3 and 4.

2.1 Overview of the DSAP and MRF for Predictive Operations

The overall DSAP Framework is built on RAM Laboratories' underlying Extensible Grid technology and utilizes Object Request Broker (ORB) concepts. This technology is open source and has been developed to support a number of defense programs involving the Air Force, Navy, and Missile Defense agency. The high level overview of DSAP is shown in Figure 2-1.

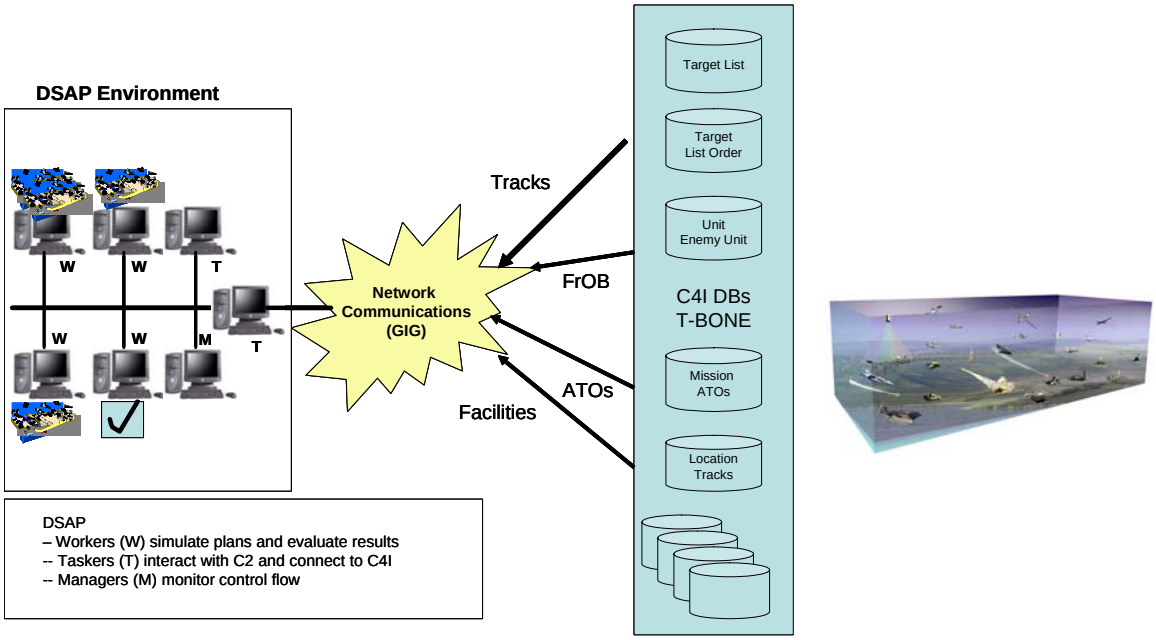

Figure 2-1: DSAP Operational View

2.1.1 DSAP Operation

DSAP encompasses two key elements: (1) dynamically assessing the operational situation and (2) dynamically predicting the outcomes of plans and alternatives based on the current situation. This section covers the dynamic prediction capabilities of DSAP.

Figure 2-2 illustrates the predictive capability of DSAP. The individual plans *y(t)* can be idealized and mapped out in time. This basically represents the behavior of the plan when the plan is executed "according to plan". Plans and their alternatives are then simulated faster-than-real-time, as denoted by the *x(t)* axis. By executing these plans faster-than-real-time, we provide a predictive look into how a plan may unfold. Multiple plans and multiple replications of each plan may be executed to provide a statistically significant outlook at a plan's anticipated

outcomes based on the current operational information. This provides the dynamic prediction capability.

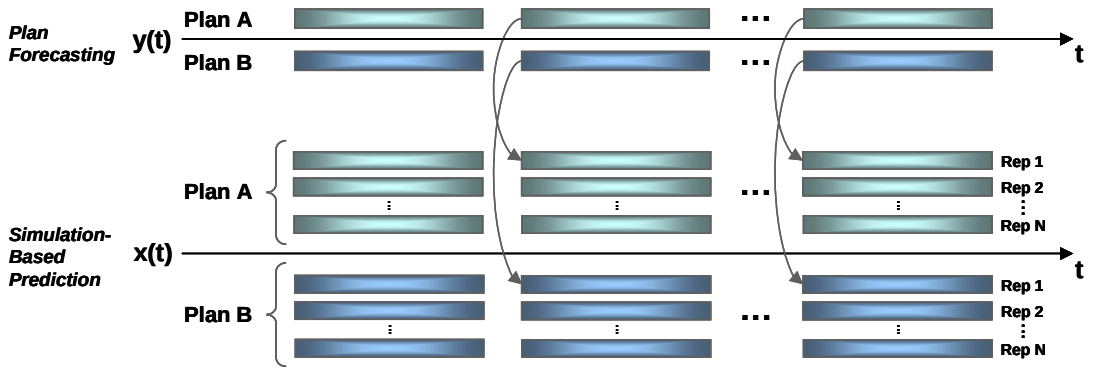

Figure 2-2: COA Simulated Faster-Than-Real-Time for Prediction

At update points, the predictive simulations, *x(t)*, are evaluated against the current real-time picture (RTP) . This evaluation process is used to rank plans, identify plans and replications that may be invalid, and prune plans and replications depending on their rank and validity.

A high level overview of the current operation of the DSAP prototype with respect to these timelines is shown in Figure 2-3. This high level overview encompasses the MRF, the simulation component (JSAF), and the real-time information updates (TBMCS). For the DSAP prototype, the MRF is used to farm-out and run multiple evaluations of both plans and alternative COAs via faster-than-real-time simulation. When real-time updates are available, real-time state information from the real-time state estimation simulation and TBMCS inputs are saved and compared with state information from the predictive plans. The plans are evaluated against each other and against the real-time picture. Plan replications that are deemed invalid (in comparison to the real-time picture) are automatically pruned and replaced. In addition, the Commander has the ability to replace plans with other alternatives based on their evaluation.

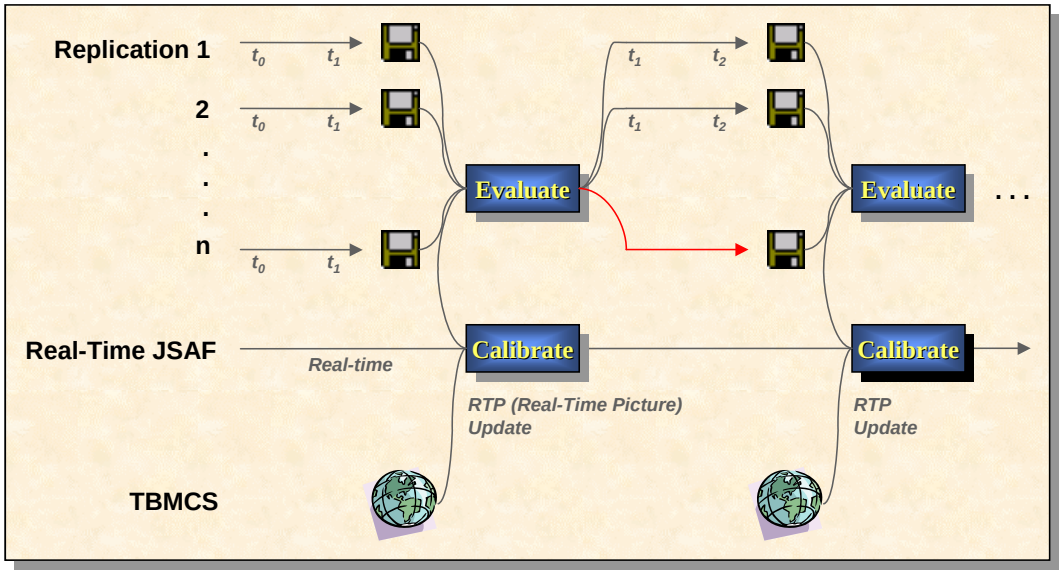

Figure 2-3: Evaluating Multiple Plans and Replications for DSAP

2.1.2 DSAP Implementation

This DSAP prototype allows multiple replications of a simulation to run concurrently across parallel and distributed platforms, gather global statistics on the simulations upon completion, splice up simulation executions into shorter runs, and compare simulation results with real-time data to prune simulations that diverge from the real-time picture. This process is managed by the MRF in conjunction with the simulation component, JSAF, and real-time information updates provided by TBMCS. The MRF architecture is shown in Figure 2-4.

2.1.2.1 Predictive Analysis

In the MRF, the *MultiRepTasker* client functions to kick off the plan evaluation process by tasking the simulation of replications of COAs and alternate COAs to available *MultiRepWorkers* running JSAF faster-than-real-time. The *MultiRepTasker* allows the user to determine which simulation scenario to run, the scenario execution name, and the start and end time of the simulation. Objectives for the COA are also specified through the use of the Console or the *MultiRepGui*. The *MultiRepManager* functions as the server in that it handles receiving, queuing, and intelligent task distribution. The *MultiRepManager* farms out replications for each plan that are run faster-than-real-time on available processors. When completed, the results of those replications are written to memory and new replications can begin in that same time slot.

An overview of the current capabilities of the DSAP prototype is shown via the MRF in Figure 2-4. The MRF serves to farm-out and run multiple replications of plans and alternative COAs via faster-than-real-time simulation. When real-time updates are available, state information from the real-time state estimation simulation are calibrated with real-time C4I inputs, saved and compared with the state information from the predictive plans. Plan replications that diverge from the real-time picture are automatically pruned and replaced. Our MRF prototype manages this entire process by utilizing TBMCS for our real-time C4I inputs and Joint SemiAutomated Forces (JSAF) as both the real-time and faster-than-real-time simulation components. JSAF was selected as our simulation component because of its ability to simulate a Joint Urban Operations (JUO) environment (as well as theater operations) as well as its enhanced support for intelligent ground clutter models, which is the current focus for the Air Force sponsor. It should also be noted that other simulations can be used as the simulation component depending on the desired application.

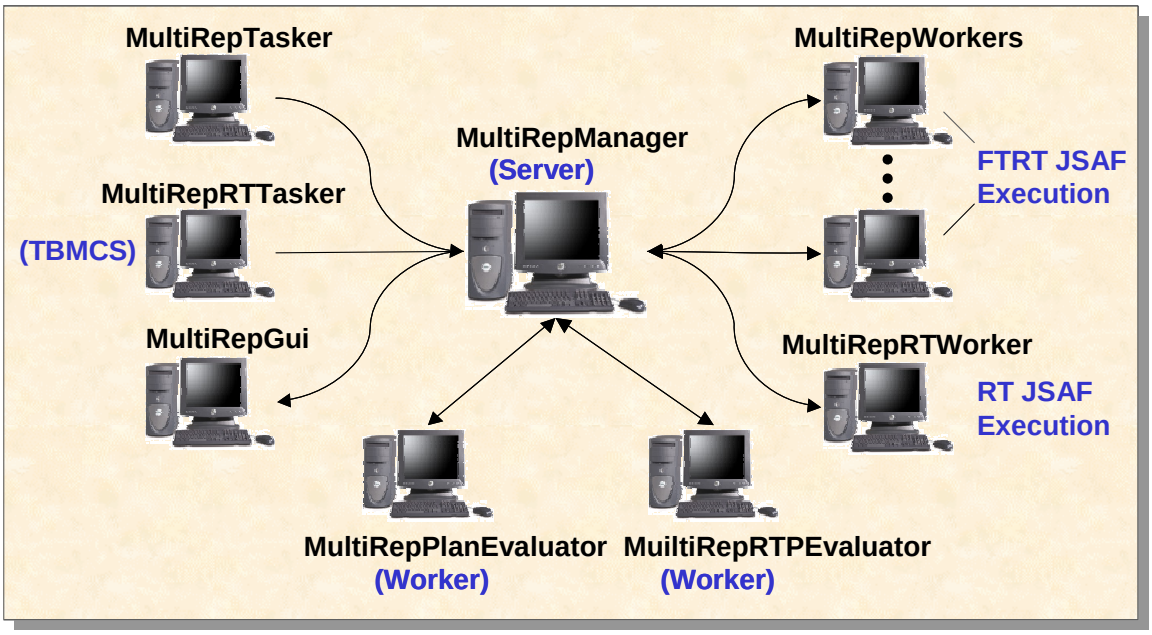

Figure 2-4: The MRF

2.2 MRF Components

The MRF contains three basic types of components: Taskers, Workers, and a Manager. Taskers function to task the manager with applications for workers to run, and specify how these applications should be initialized. Workers complete the tasks assigned by the manager, including executing the simulation replications, saving the results of the replications, and evaluating and comparing the results of the replications to the plan objectives and real-time picture. The manager divides the replications into smaller time segments and assigns these tasks to the workers, handles the bookkeeping, and tackles flow control issues. Figure 2-4 illustrates each of these components and their connectivity with the manager. The role of each of the specific MRF components is discussed in the following subsections.

2.2.1 MultiRepTasker

The *MultiRepTasker* component interfaces with Command and Control to send a predictive simulation task to the server. The *MultiRepTasker* provides connectivity to the server and allows the user to specify initialization parameters such as the simulation execution name, initial scenario file, start and end time, simulation scaling rate, and replication number.

2.2.2 MultiRepRTTasker

The *MultiRepRTTasker* component issues a task to the server to initiate the real-time worker. The *MultiRepRTTasker* provides connectivity to the server and allows the user to specify the simulation execution name, initial scenario file, name of the plan the task corresponds to, and the time interval between saving the state of the simulation.

The Real-Time Tasker component, *MultiRepTBMCSTasker*, provides the capability to allow the user to retrieve the Real-Time Picture (RTP) from TBMCS or another C4I data source via command line or GUI. The sequence diagram defining the operation of the *MultiRepTBMCSTasker* is shown in Figure 2-5.

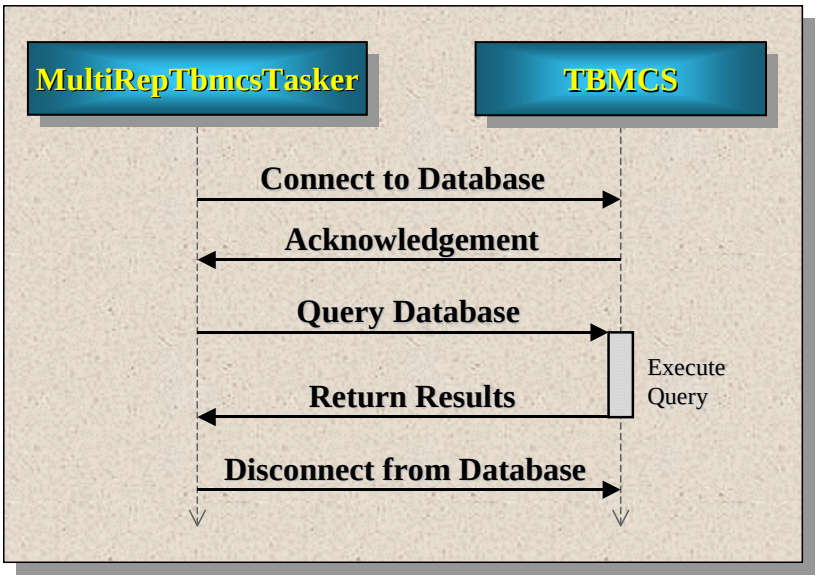

Figure 2-5: Sequence Diagram for MultiRepTBMCSTasker and TBMCS:

The *MultiRepTBMCSTasker* is a Tasker component that automates the process of retrieving real-time C4I information from TBMCS. For TBMCS connectivity, both ODBC and JDBC were tried. Some problems existed with ODBC and have not been resolved. The Tasker/JDBC approach to TBMCS connectivity has been implemented and tested.

2.2.3 MultiRepManager

The server, or *MultiRepManager*, component is the core of the MRF. The *MultiRepManager* is responsible for 1) managing the execution of long replications by splicing them in time, 2) constructing the necessary parameters needed for a worker to launch and save a JSAF execution, 3) constructing the necessary parameters for launching an evaluation on an evaluator component, 4) displaying diagnostics related to the execution of multiple replications, 5) identifying when replications are completed, 6) pruning and re-tasking replications that are off course from the real-time picture, and 7) restarting unfinished replications in the event of a worker crash or disconnect.

The *MultiRepManager* design builds off of the *WpServer* used to implement the Extensible Grid. The Server capability for this effort inherits from the WpNetGridServer, which is the server for the Extensible Grid, which in turn inherits from WpServer, which is the basic server capability. The UML Class Diagram for the Server design is shown in Figure 2-6.

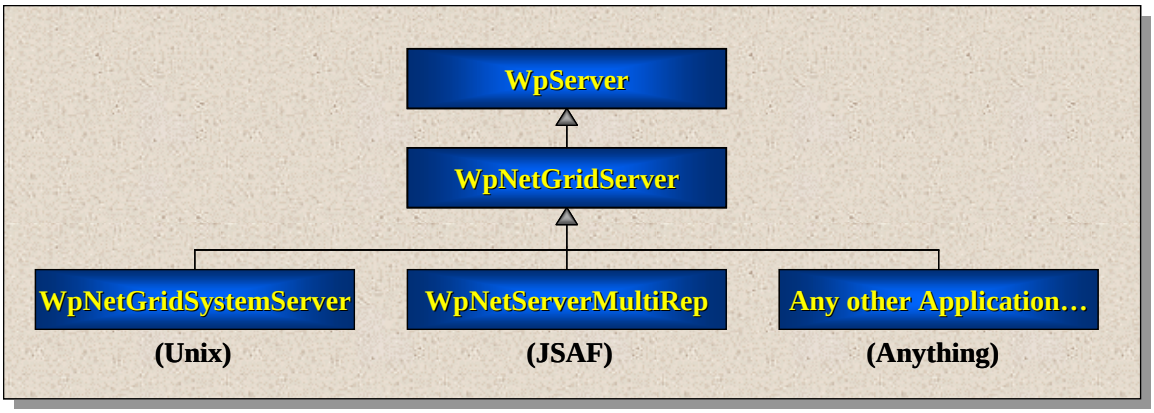

Figure 2-6: UML Diagram for Server Component Used for the MultiRepManager

The *MultiRepManager* is responsible for managing the execution and evaluation of the replications. The Activity Diagram for the *MultiRepManager* with respect to the *ProcessRtp()* function is shown in Figure 2-7. This Activity Diagram defines the process for managing the replications through their RTP evaluation.

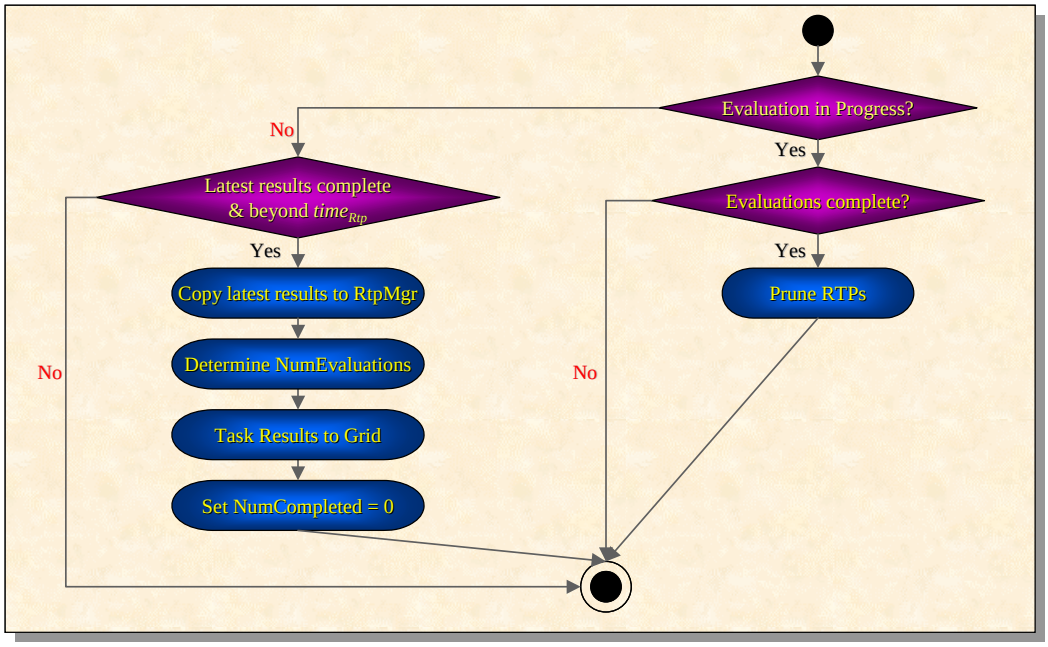

Figure 2-7: Activity Diagram for MultiRepManager's ProcessRtp()

2.2.4 MultiRepGui

The *MultiRepGui* component provides the user with diagnostics with respect to operation of the MRF. The *MultiRepGui* also allows the user to monitor the status of the MRF. The Sequence Diagram for the *MultiRepGui* is shown in Figure 2-8. In addition to simply monitoring status and setting the time interval for faster-than-real-time simulations, the *MultiRepGui* has been modified to host our GUI. The *MultiRepGui* now queries the server to return the status of replications, provides functionality to modify time intervals and end times, provides functionality to modify pruning thresholds, and provides the capability to allow the user to prune replications or plans using the GUI.

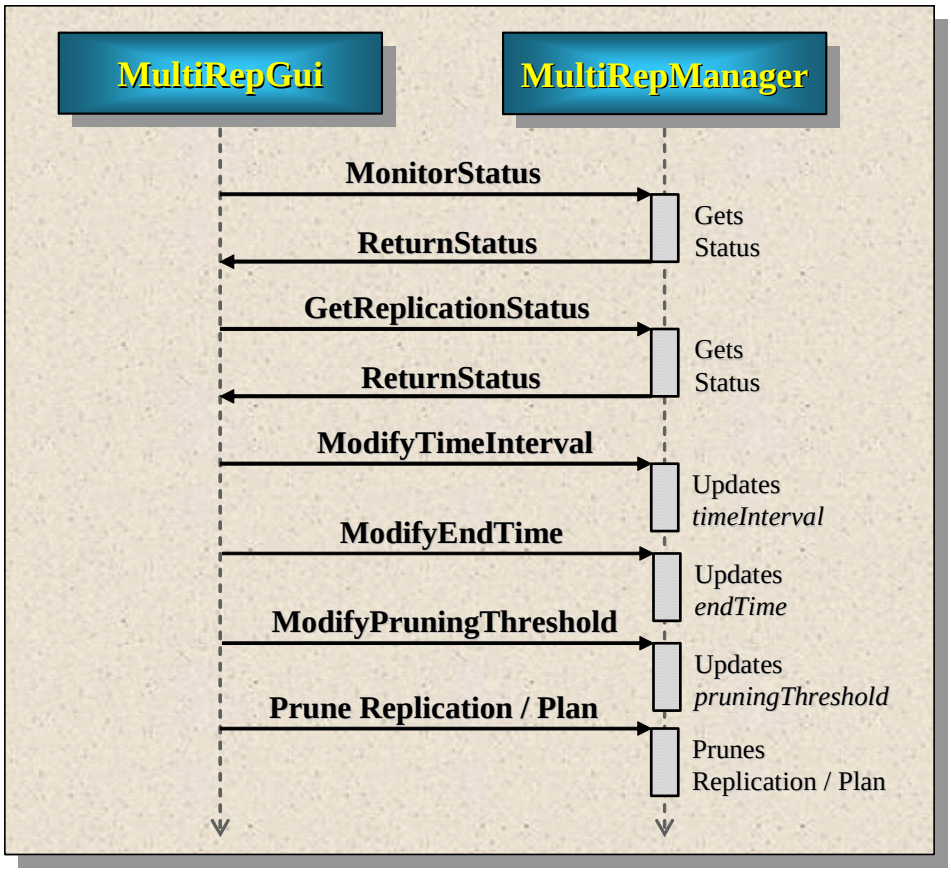

Figure 2-8: MultiRepGui Control Flow

2.2.5 MultiRepWorker

The Worker component, *MultiRepWorker*, receives simulation tasking from the *MultiRepManager* and launches predictive JSAF executions that run faster-than-real-time. The Worker is responsible for launching JSAF replications faster-than-real-time for a predetermined length of time. This command is accompanied by parameter sets (specifying the *SimRate*, start time, end time and other variables), environment variables and scenario spreadsheets. The Worker also packs up results from the replication execution in spreadsheet format and sends the information back to the *MultiRepManager*.

Upon completion of the replication, the Worker saves the state of the simulation to disk and sends it to the server for later evaluation and comparison with real-time data and the plan objectives. Replications that stray from the real-time picture are automatically pruned, re-tasked by the server, and initialized to match the current state. Replications that fail to meet the plan objectives can be manually pruned by Command Staff, and if pruned, they are automatically re-tasked by the server and initialized to match the current state.

The Sequence Diagram specifying the operation of the Worker executing faster-than-real-time JSAF scenarios is shown in Figure 2-9 with respect to the rest of the MRF. This is the heart of the predictive capability for DSAP/MRF.

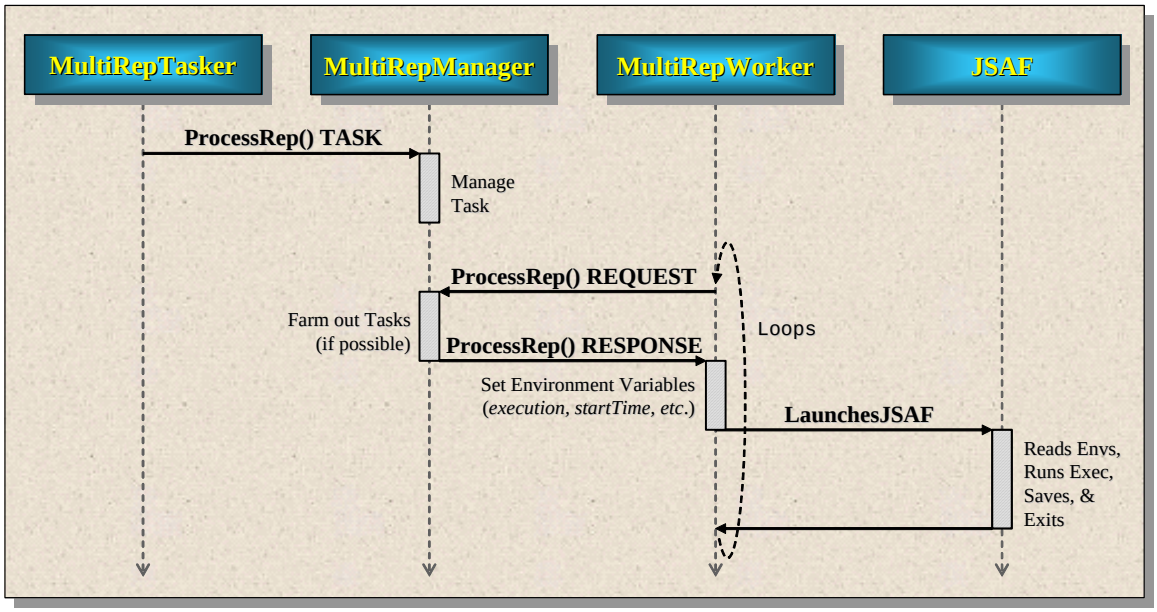

Figure 2-9: MultiRepWorker Control Flow for Executing Faster-Than-Real-Time Simulations

At update time boundaries, the real-time simulations running on *MultiRepWorker* components are synchronized with the real-time picture (in this case, live or emulated data from the Theater Battle Management Core System's (TBMCS) MIDB and AODB, which is updated every fifteen minutes via subscription). The *MultiRepRTTasker* or *MultiRepTBMCSTasker* client feeds real-time information from TBMCS to the system and initiates the update process.

2.2.6 Plan Evaluator

The Plan Evaluator component, *MultiRepPlanEvaluator*, is responsible for comparing the state of the saved faster-than-real-time replications with the plan objectives. The *MultiRepPlanEvaluator* is a Worker that evaluates the results of the JSAF replication executions against other results. The *MultiRepPlanEvaluator* takes each of the result spreadsheets and evaluates them to determine the "best" plan. The evaluation is performed by executing the function *PlanEvaulator()*. The control flow for the *MultiRepPlanEvaluator* is shown in Figure 2-10, when considering the sequence of operations between the *MultiRepPlanEvaluator*, *MultiRepManager*, and *MultiRepWorkers*. The *MultiRepPlanEvaluator* requests tasks from the server. When results spreadsheets are available at the *MultiRepManager*, those results are tasked to the *MutliRepPlanEvaluator*, which evaluates the effectiveness of each plan. The effectiveness results are then sent back to the *MultiRepManager*, and the *MultiRepRTPEvaluator* is also tasked to begin evaluator those results against the current real-time picture.

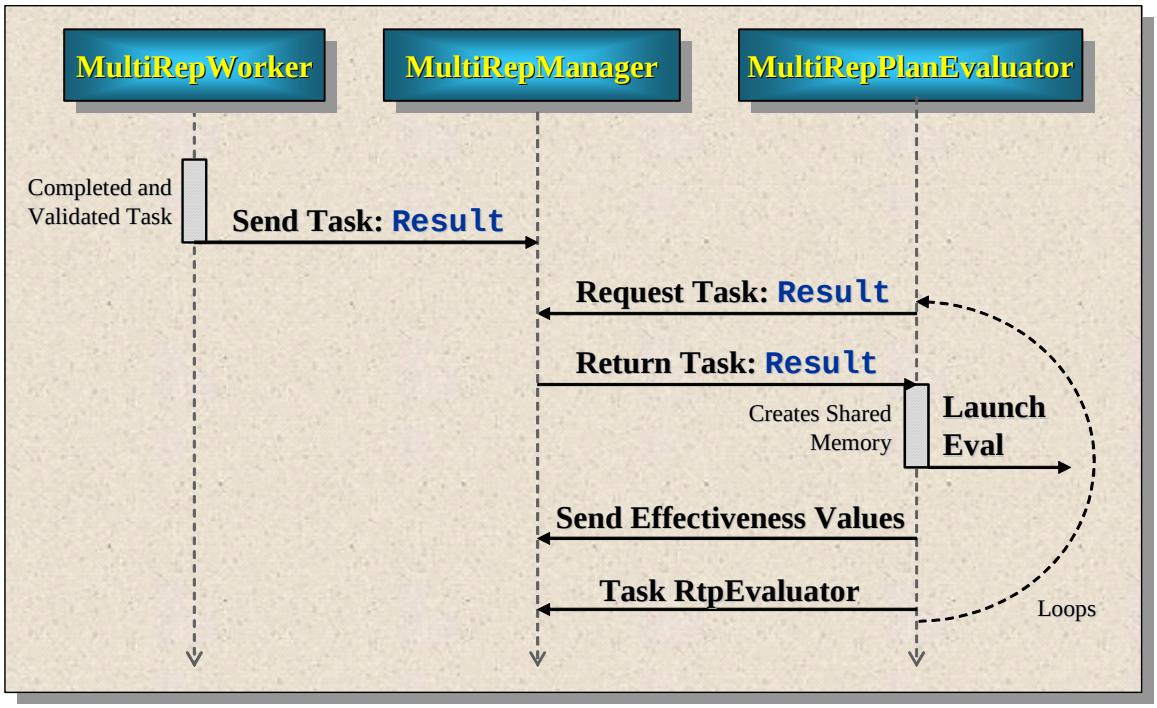

Figure 2-10: Control Flow for MultiRepPlanEvaluator

Workers (*MultiRepPlanEvaluator* and *MultiRepRTPEvaluator*) are used to evaluate the predictive simulation executions with respect to both their objectives and the real-time picture. The evaluation process computes both the *Raw Effectiveness* and the *Relative Effectiveness* of each replication. These Measures Of Effectiveness (MOEs) are used to rank each COA and determine if each plan is consistent with the real-time picture. An alternative plan may be selected or the existing COA may be maintained depending on its effectiveness. Also, if the current real-time picture shows that an alternative COA is no longer valid, that COA can be pruned or replaced by a valid alternative COA. Scenario data within the MRF's evaluation process is also updated to reflect changes in assets and resource status based on real-time picture updates. At that point, the COAs and alternates are once again farmed out and executed on the available processors.

The MRF GUI is shown in Figure 2-11. The GUI provides a graphical interactive interface to the MRF that allows Commanders to task the execution of simulation plans and replications, view the progress and performance of the plans, and prune ineffective plans. The graph at the bottom of the GUI plots the raw and relative effectiveness of each plan over time. These metrics are used to gauge the effectiveness and performance of the Commander's plan.

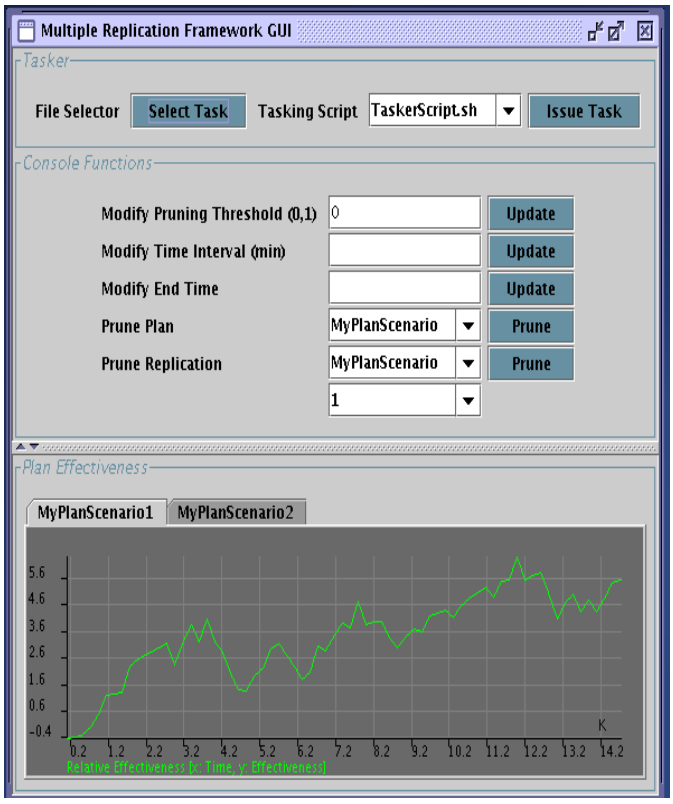

Figure 2-11: The MRF GUI

Plans and replications are initiated in the MRF by issuing a tasking script from the GUI. As shown in Figure 2-12, a file selector tool allows the Commander to select a tasking script to kick off the process. The GUI will be expanded to allow the Commander to start plans and replications via menus instead of scripts.

After the script has been selected, the MRF takes control by sending the task to available workers, as shown in Figure 2-13.

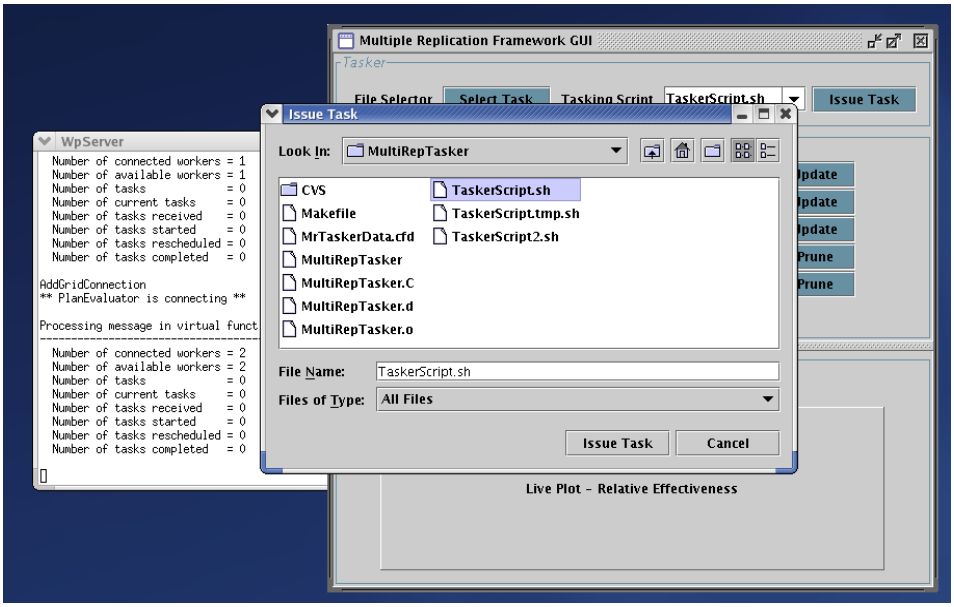

Figure 2-12: Selecting and Issuing a Tasking Script

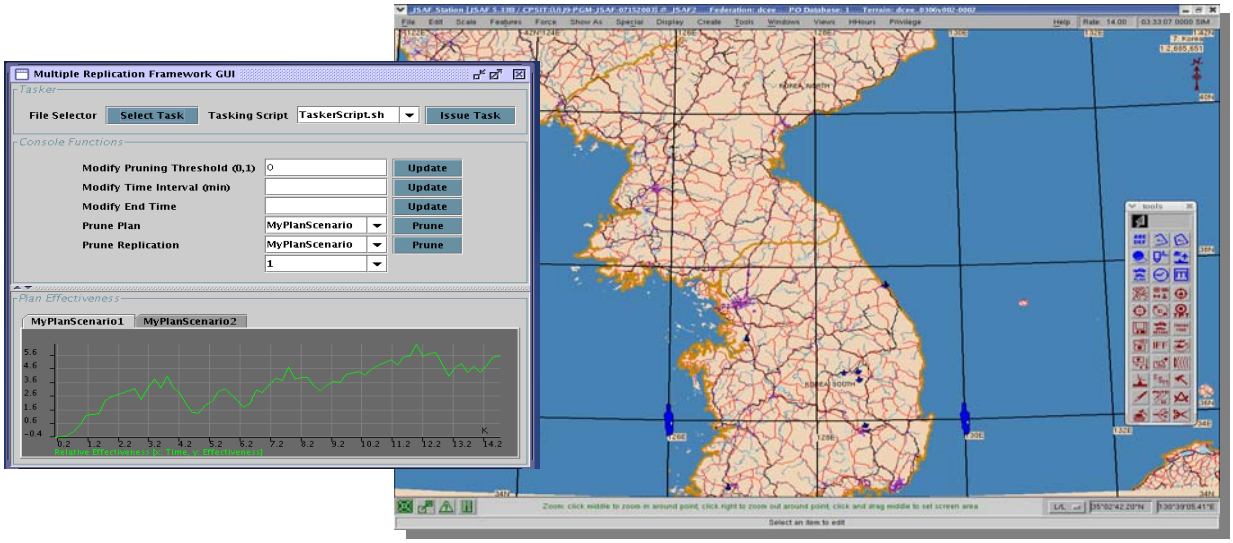

Figure 2-13: GUI kicking off a JSAF Replication

After the simulation time segment is complete, the MRF automatically performs the plan evaluation and real-time picture evaluation, if the real-time data update was received. Figure 2-14 shows the plan evaluation results of a replication.

Because of the uncertain nature of predicting the outcome of plans, it is important to execute multiple replications of a plan and statistically analyze the results. The MRF calculates the mean and standard deviation of the effectiveness values for each time step in the simulation. These mean values and their standard deviations can be fitted using $\chi 2$ analysis to obtain time-based curves that provide trend analysis. The $\chi 2$ analysis is used to compare both the simulated and observed results with the expected results of the plan.

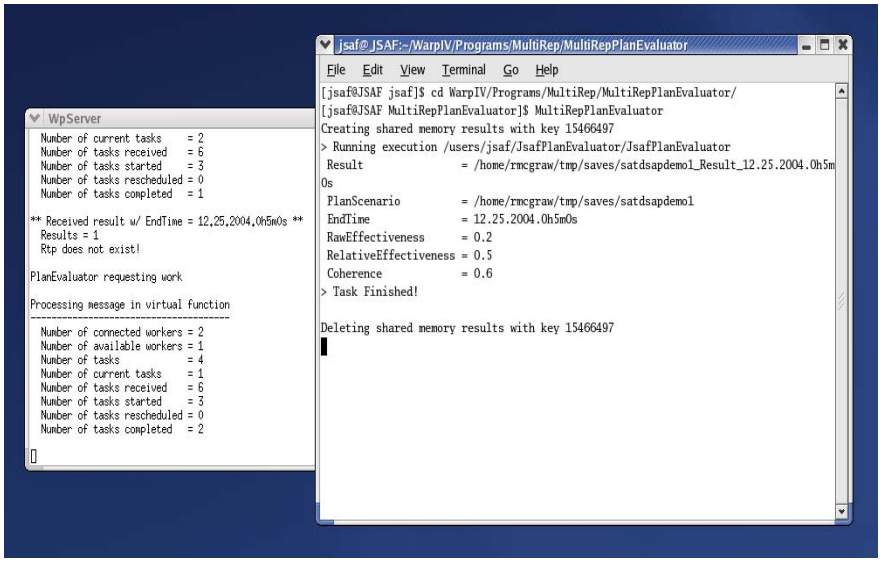

Figure 2-14: Plan Evaluation Results

3.0 Modifying the DSAP MRF for Dynamic Situational Awareness

This effort focused on providing the real-time dynamic situational awareness capability for operational C2 through the use of embedded simulation. The earlier DSAP work had focused on implementing the predictive analysis component. This effort focused on implementing the dynamic situational awareness component that utilized both a real-time simulation component and a real-time calibrated simulation component. This section discusses the modifications that were made to the DSAP framework to provide these capabilities.

3.1 DSAP Concept of Operations for Dynamic Prediction

The DSAP operational concept describes how the DSAP capability can be applied for dynamic situational awareness. Figure 3-1 shows the evolution of several plans with respect to time. The $x_p(t)$ axis shows the current plan being simulated in real-time. The real-time simulation, in our case JSAF, tracks the real-time operational picture, while updating Blue and Red Force information via real-time updates. Real-time updates, denoted by the $z(t)$ axis, are provided to the real-time simulation of the current plan. These updates are used to correct the predicted behavior of the simulated COA. The updated simulation can be checkpointed to save and estimate the state of the real-time operational picture. This is denoted by $x_e(t)$. This allows us to store the internal state, $x_e(t)$ of the mission in a manner that will augment the information provided by external "visible" behaviors. This provides us with our dynamic situation awareness capability.

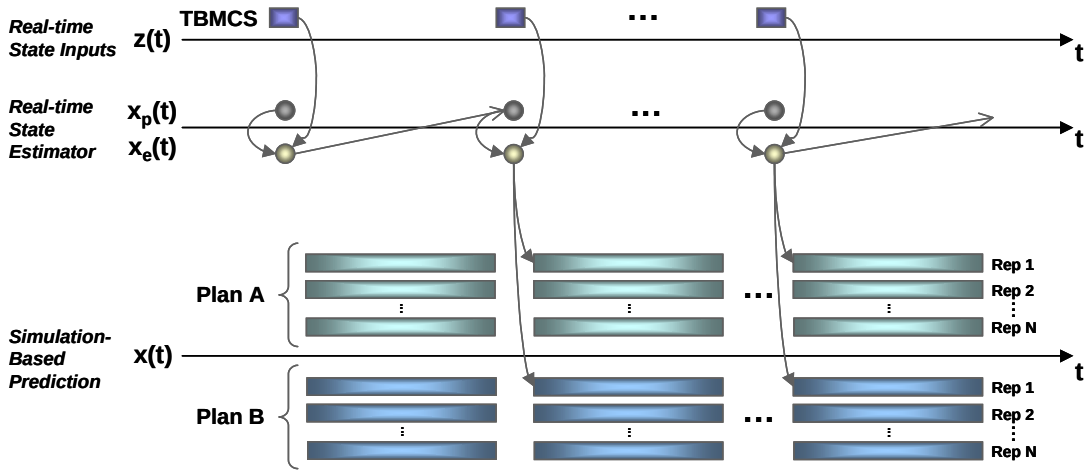

Figure 3-1: Real-time Simulation and Updates for State Estimation

3.2 Updates to the MRF

The MRF discussed in Section 2.0 provides a predictive analysis capability for C2 through the use of embedded simulations. As mentioned, the MRF was built on RAM Laboratories Extensible Grid Architecture and our underlying Object Request Broker technology. The MRF implemented in support of predictive operations consisted of the following components: a *MultiRepTasker* to kick off simulation and analysis runs, a *MultiRepTBMCSTasker* to pull-in data from TBMCS' AODB and MIDB, *MultiRepWorkers* to execute simulations using JSAF or other simulation environments of choice, *MultiRepPlanEvaluators* to evaluate predicted plan

effectiveness, *MultiRepRTPEvaluators* to evaluate predicted plan effectiveness in comparison with the TBMCS updates, and the *MultiRepManager* which was used to maintain the control flow and monitor the flow of data through the MRF.

For this effort, the MRF was modified to provide a real-time dynamic situational awareness capability. In this manner, the MRF simulated the current plans in real-time. Two real-time simulations were supported: a real-time simulation of the idealized plan, and a real-time simulation that was constantly calibrated with TBMCS and eventually TBONE inputs. The modifications to the MRF architecture, topology, data flow and sequence of operations are discussed in the following sections.

This effort updated the MRF as shown in Figure 3-2. Here, a Real-time Worker (*MultiRepRTWorker*) and a Real-Time Calibrated Worker (*MultiRepRTCWorker*) are added to the framework. The *MultiRepRTWorker* component is responsible for executing the idealized simulation of the plan using the simulation environment of choice (in our case JSAF). This simulation is not calibrated with real-time inputs. Its intent is to play out the idealized plan based on the state of operations at the plan initiation. The simulated state of operations for the idealized plan is saved and sent to the *MultiRepManager* every user-defined time segment. Our implementation uses JSAF's native damage reporting capability to store the plan state in comma delimited format. This idealized plan information is used to compute Real Effectiveness by the *MultiRepRTPEvaluator*.

The second component implemented on this effort is the real-time calibrated worker (*MultiRepRTCWorker*). This component receives real-time updates, and must shut down and restart to receive and reflect the updates. This process was implemented on this effort using JSAF's checkpoint and restarted capabilities. Additional work was performed to calibrate this worker (and thus the JSAF simulation) without utilizing this checkpoint restarted process. This approach would allow for the worker to receive and reflect the updates while the simulation is running via sockets, shared memory or additional approaches.

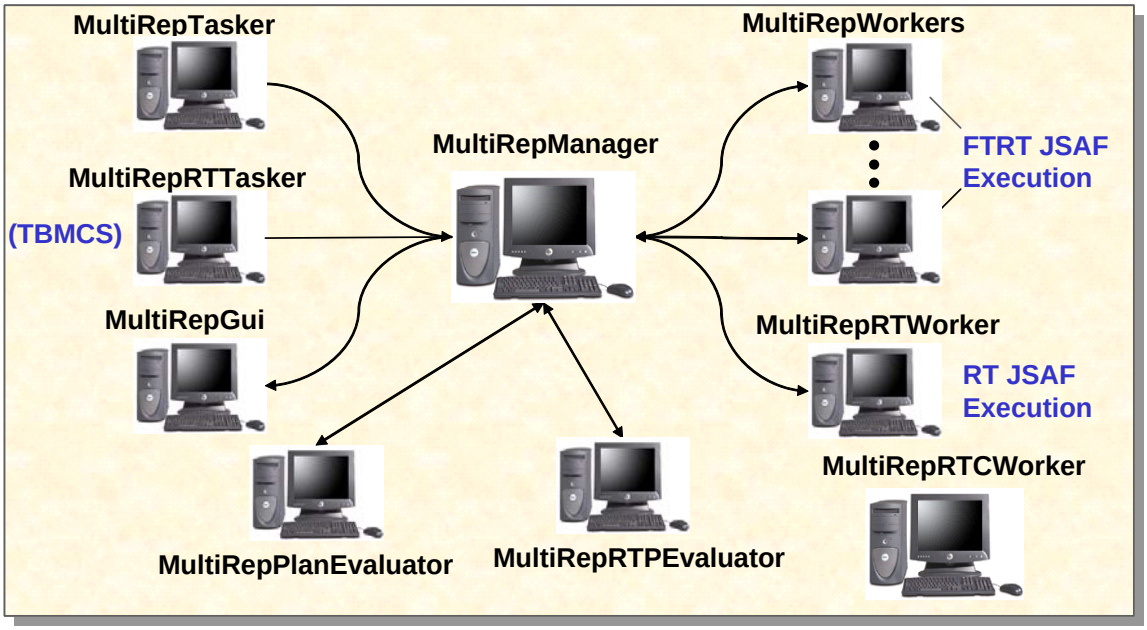

Figure 3-2: MRF Updates for Dynamic Situational Awareness Capability

3.2.1 Updates to the MultiRepRTPEvaluator

For DSAP's predictive operations, our approach used a *MultiRepRTPEvaluator* that compared the state of predictive plans to the current real-time picture to ensure that the predictive analyses were consistent with the current state of operations. In support of the Dynamic Situational Awareness capability, this *MultiRepRTPEvaluator* was modified to ensure that the RTP Evaluator compared the idealized plan (simulated on *MultiRepRTWorkers*) with the calibrated real-time plan (simulated on *MultiRepRTCWorkers*). This approach resulted in several updates to the control flow associated with the *MultiRepRTPEvaluator*. The first change, shown in Figure 3-3, added functionality for looping through RTP evaluation results, and pruning replications that exceed some (user-specified) threshold. This modified control flow also assumes that the *MultiRepRTPEvaluator* has connected to the server. The second modification to this control flow, shown in Figure 3-4, shows the modified control flow based on changes made to the *MultiRepTasker* components which combined the functionality of the emulated TBMCS and TBMCS real-time calibration process.

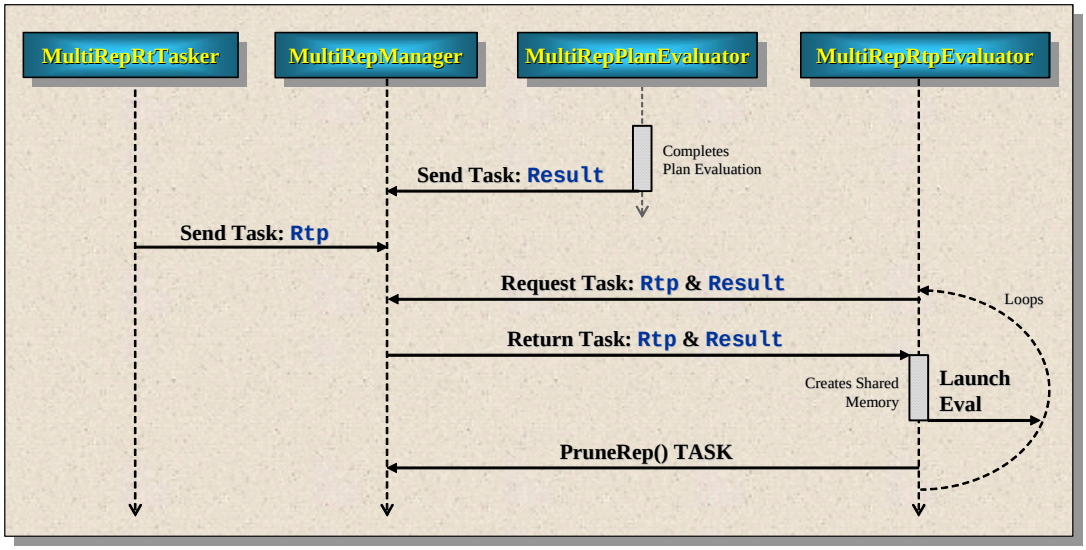

Figure 3-3: RTP Evaluator Modification to Prune Replications

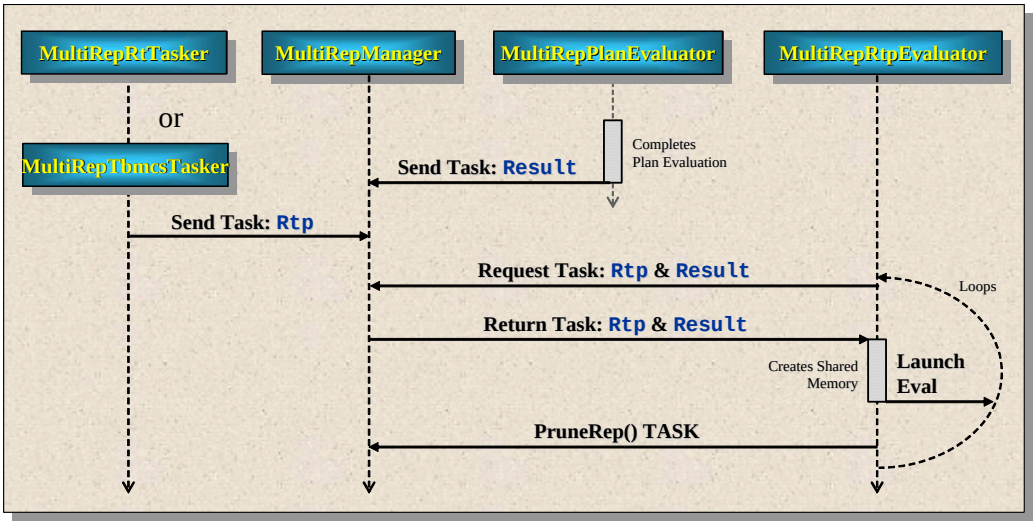

Figure 3-4: RTP Modification to Reflect Tasker Enhancements

3.2.2 MultiRepRTCWorker

The *MultiRepRTCWorker* is used to run real-time JSAF executions (or executions of other simulations) in order to support the state estimation capability in the MRF. The *MultiRepRTCWorker* is responsible for running the real-time simulation, updating the simulation with TBMCS information (in the case of the calibrated real-time simulation), and saving checkpoints of the simulation to intermediate results files for effectiveness evaluations. The sequence diagram of the *MultiRepRTCWorker* is shown in Figure 3-5. For the *MultiRepRTCWorker*, the worker connects to the server and requests tasking. The Tasking, when provided to the server from the *MultiRepRTTasker* (the old *MultiRepTBMCSTasker*) then assigns tasks to the server, which are passed to the *MultiRepRTCWorker*. The *MultiRepRTCWorker* executes JSAF tasks and sends results back to the Server every 15 minutes. In addition, the *MultiRepRTTasker* continually updates the real-time simulation every 15 minutes.

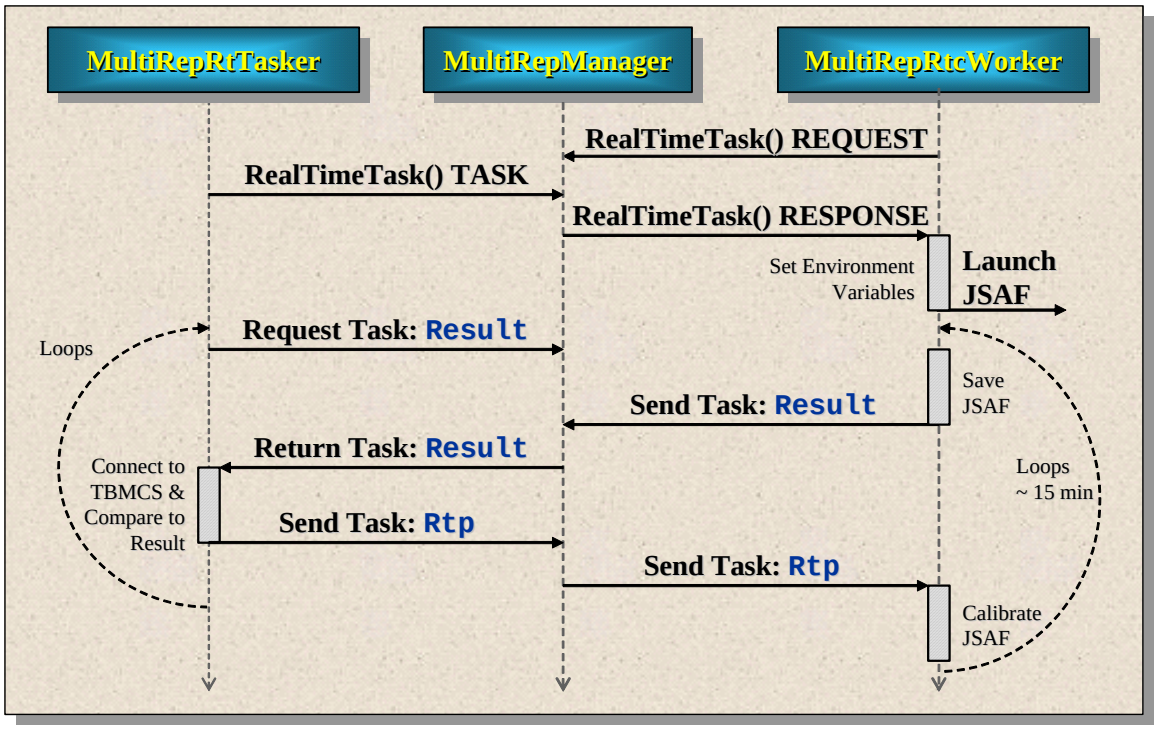

Figure 3-5: Sequence Diagram for MultiRepRTCWorker

3.2.3 MultiRepRTWorker

The *MultiRepRTWorker* component simply executes the idealized plan in real-time. The *MultiRepRTWorker* behaves like the *MultiRepRTCWorker* and *MultiRepWorker* with the exception that the *MultiRepRTWorker* does not calibrate with real-time data. The *MultiRepRTWorker* saves its state every 15 minutes and sends this information to the server. The *MultiRepRTWorker* remains up and does not close during the execution of the MRF. The sequence diagram of the *MultiRepRTWorker* is shown in Figure 3-6.

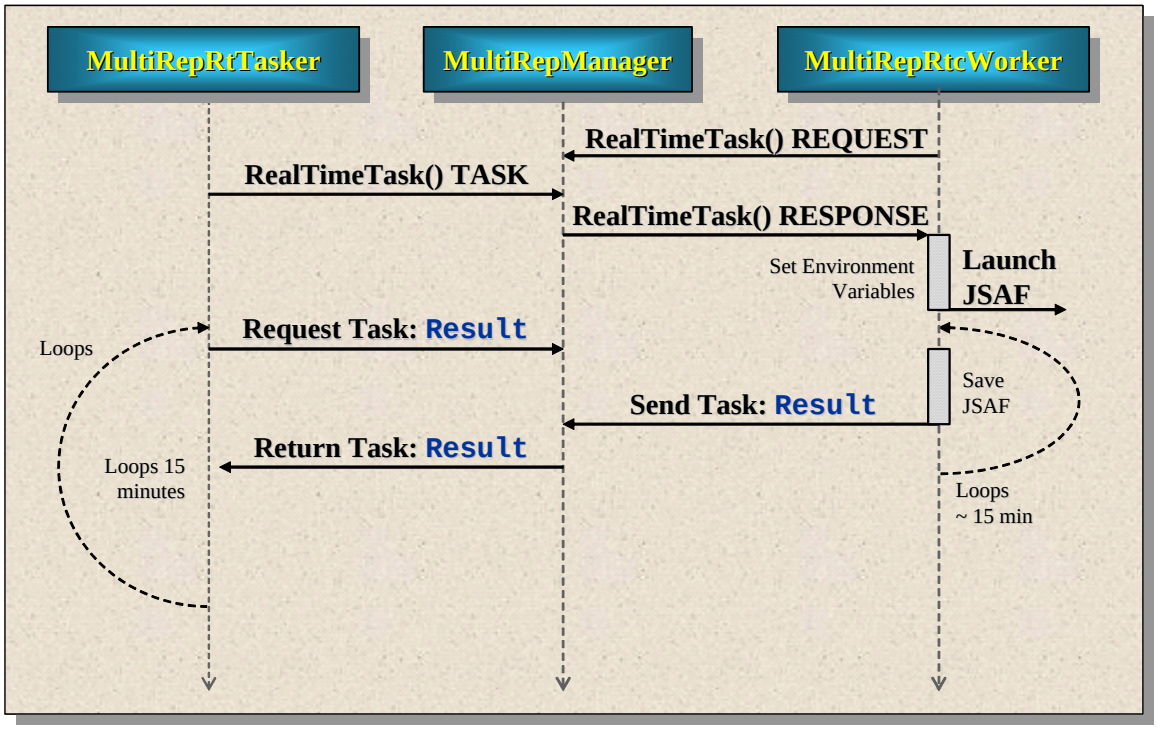

Figure 3-6: Sequence Diagram for MultiRepRtWorker

Once these components were implemented, the control for this dynamic situation assessment capability was defined and built within the MRF to handle the Real-Time Worker and Real-Time Calibrated Worker components. The sequence diagram for these capabilities is shown in Figure 3-7.

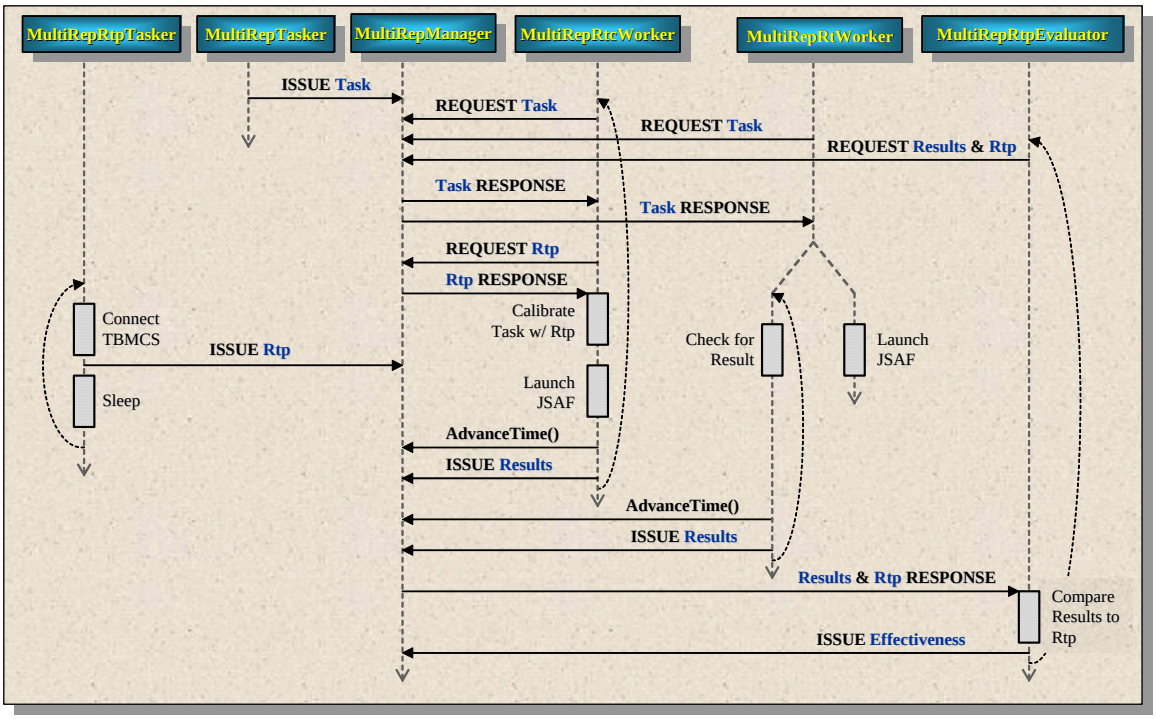

Figure 3-7: Dynamic Situation Assessment Sequence Diagram.

3.3 Real-time Extensions

Extensions are being designed for the MRF to better address the extraction and calibration process using real-time C4I data. These extensions involve pulling in Route Planning capabilities and pulling in real-time data feeds for Blue-Force tracking. These are depicted in Figure 3-8 and Figure 3-9.

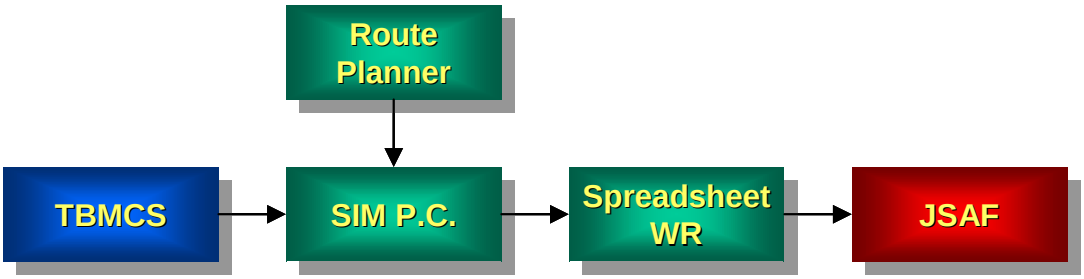

Figure 3-8: Incorporating Route Planning in the MRF

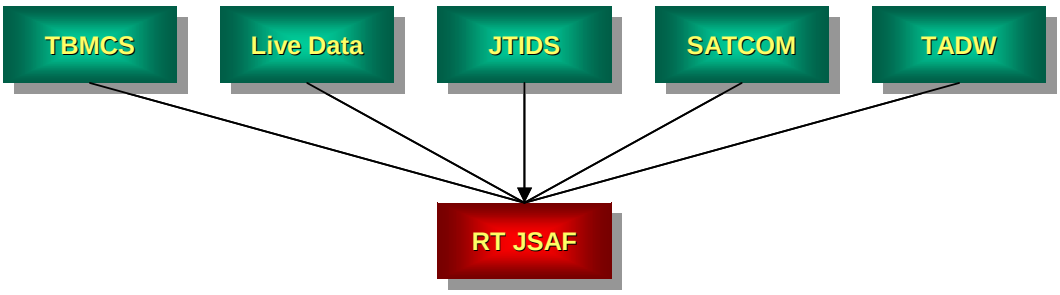

Figure 3-9: Incorporating Real-time Data Feeds for Providing Blue Force Information

3.4 Modification of Objects in MRF/JSAF

Functions were implemented to allow simulations to automatically create and modify objects in JSAF using the DSAP framework. Three approaches were examined in this regard: the use of sockets, the use of shared memory, and the use of checkpoint restart functionality. Each of these is described in detail below along with an analysis concerning the approach.

3.4.1 Checkpoint/Restart Approach

A checkpoint restart approach was initially implemented that allowed for modifying simulation state. This approach extended upon the implementation used to provide the faster-than-real-time predictive analysis capability in the previous effort. In this approach, JSAF (or other simulations) simulations were checkpointed at a user-defined point in time. This checkpointing approach used JSAF's native checkpointing capability to write the simulation state to a binary checkpoint and/or a comma delimited Excel file. The checkpoint data could then be updated or calibrated with real-time information from TBMCS or TBONE by reading-in comma delimited (or other formats) files upon JSAF's normal initialization process.

For binary checkpoints, the simulation restarted could restart from the binary checkpoint and then overwrite, delete or add information specified in the comma delimited initialization file. Drawbacks to this approach was that the simulation had to be restarted from the binary checkpoint. This process was time consuming (some benchmarks from the previous effort showed that this process took around 90 seconds for the Korea scenario) and affected the ability

of DSAP to scale for faster-than-real-time simulations and hindered the user/analysts ability to use DSAP to evaluate plans and COAs in smaller time segments.

For comma delimited checkpoints, the checkpointed files could be augmented by DSAP and the MRF and used to initialize JSAF from the updated or calibrated simulations. This approach also required a timely checkpoint and startup process. This approach resulted in "missing" and incomplete operational state data that was not stored in the checkpoints.

3.4.2 Shared Memory Approach

The second approach that was investigated for calibrating or updating simulation runs was the use of shared memory. In this approach, simulation state variables on running JSAF simulations were stored in shared memory on computing platforms where the *MultiRepWorker* also had access to that memory segment. In this case, the shared memory could be updated by the *MultiRepWorker* with calibration information from TBMCS or TBONE without having to checkpoint and restart the simulation. This approach enabled the update of simulation values directly in memory.

Drawbacks to this approach were that the user/developer needed to know exactly where, in memory, the simulation state variables were stored. This process was very dependent upon scenario size and complexity and required a new solution for each scenario being executed.

3.4.3 Sockets Approach

The third approach considered the use of Sockets for updating the running simulation. In this approach, sockets could be opened for a JSAF simulation running on a *MultiRepWorker* (or a *MultiRepRTCWorker*). The simulation internal state could be updated by executing native JSAF calls through this socket interface. Few drawbacks were apparent using this approach. The approach did not require a checkpoint and restart process and scaled more readily than the shared memory approach.

3.4.4 Summary of Calibration Methods

An analysis of these implementations has shown that the socket-based approach is the most effective at modifying the PO (Persistent Object) databases. There are some issues that are currently being examined that entail synchronizing the running simulation with the modified PO database. These are still being worked out.

3.5 Extracting Damage Results

While this effort developed processes for calibrating real-time simulations using TBMCS or TBONE data, the effort also took steps to store damage and operational state information for the running JSAF simulations to use in the Raw and Real Effectiveness calculations determined by the *MultiRepPlanEvaluator* and *MultiRepRTPEvaluator* components. The code added to the JSAF source code is highlighted in the snippet box below.

```
void SaveUnitStatus(char *fileName, PO_DATABASE *po_db) {

  if (!po_db) {
    cout << "ERROR: po_db does not exist!" << endl;
    return;
  }

  ofstream outFile(fileName);
```

```
  if (!outFile) {
    cout << "ERROR: file " << fileName << " could not be opened!" << endl;
    return;
  }

  cout << "Saving Unit Status..." << endl;
  int i;
  PO_DB_ENTRY *entry;
  time_pause(po_db->save_load_pause_handle);
  outFile << "CALLSIGN,UNIT TYPE,POSITION,LAT,LON,Z,TOTAL,CAPABLE,DAMAGED,DESTROYED"
          << endl << endl;

  for (i=0; i<PO_MAX_OBJECT_CLASS; i++) {
    for (entry = po_db->object_classes[i]; entry; entry = entry->next_class) {
      if (!entry->implied && !entry->deleted) {
        string callsign = (char *)PO_UNIT_DATA(entry).marking.text;
        string type = stdname_get_standard(
              protocol_find_name(PO_UNIT_DATA(entry).objectType));
    if (!callsign.empty() && type != "Constant Unknown") {

      struct usg_sub_counts usc;
      usc.total = 0;
      usc.capable = 0;
      usc.damaged = 0;
      usc.destroyed = 0;

      unitorg_traverse_tree(entry,
                  TRUE,
                  UsgAddVehicle,
                  (UNITORG_FUNCTION_ARG)&usc);

      LatLonCoordinates ll = GcsToLatLon(PO_UNIT_DATA(entry).location);

      outFile << callsign << ","
            << type << ","
            << UsgGetLocString(PO_UNIT_DATA(entry).location) << ","
            << ll.Lat << ","
            << ll.Lon << ","
            << ll.Z << ","
            << usc.total << ","
            << usc.capable << ","
            << usc.damaged << ","
            << usc.destroyed << endl;
    }
   }
  }
 }
```

Code Segment 3-1: Code for Extracting Damage and State Results from Running JSAF Simulations

3.6 Summary

This section has detailed the modifications and enhancements performed on this effort to DSAP's MRF to support the real-time Dynamic Situational Awareness capability for operational C2. This effort developed real-time simulation components for *MultiRepWorkers* to support idealized and calibrated simulations, re-defined the control flow for real-time dynamic situational awareness applications, implemented mechanisms to extract damage and operational state data from running real-time simulations, and implemented mechanisms to calibrate real-time simulations from TBMCS or TBONE inputs.

4.0 Extending DSAP for a Service Oriented Architecture

The DSAP Framework prototype has been implemented in a grid-computing environment using Object Request Broker technology. This infrastructure allows available processors within the environment to perform work and generate tasks as their time becomes available. The implementation interacts with real-time databases via a classic 3-tiered architecture by subscribing to information updates in TBMCS and TBONE. One of the major goals of this effort was to initiate the process of making the DSAP Framework "GIG-ready."

One of the overarching goals of this effort was to start extending the DSAP Framework to support Network Centric Operations and Warfare (NCOW). To address this goal, the DSAP Framework is being enhanced to work with Network Centric Enterprise Services (NCES) in a Service Oriented Architecture in order to support applications utilizing the Global Information Grid (GIG). The starting point for ensuring that DSAP supports the GIG was to ensure that elements of the infrastructure provide a publish/subscribe capability. This publish/subscribe capability enables processing nodes to (1) publish data or information, (2) subscribe to published data or information, and (3) support query on demand operations for posted data. The operation of the DSAP Framework in this context is described as follows and is depicted in Figure 4-1.

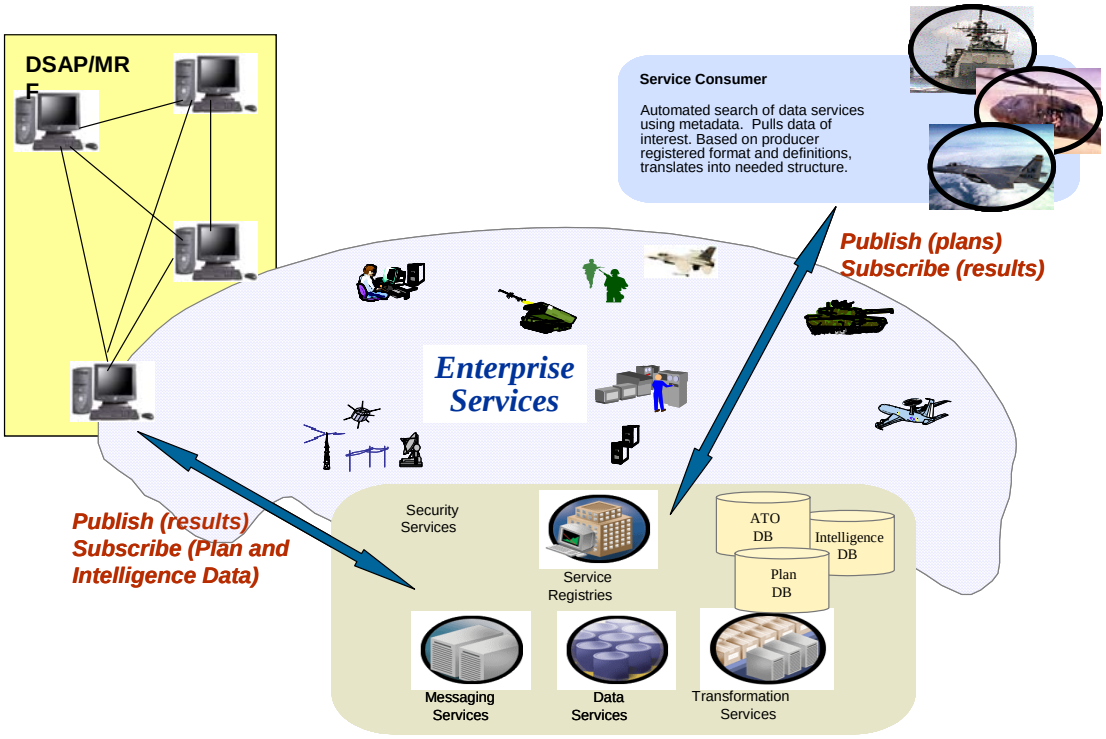

Figure 4-1: Enhancing DSAP for a Service Oriented Architecture

The underlying communications infrastructure of the DSAP framework was augmented with publish/subscribe functionality. While providing these underlying mechanisms for Network Centric Operations, additional work was performed to define the data and services that the DSAP Framework will provide to the GIG community.

For example, in their paper on M&S in the GIG environment presented at the IITSEC Conference in 2004, Numrich, Hieb, and Tolk present a view for supporting NCOW in a GIG environment. Among the elements discussed is a value chain based around the concepts of Data Quality, Information Quality, Knowledge Quality, and Awareness Quality. The DSAP Framework can be viewed as a tool for supporting NCOW because it provides measures to support Data Quality (through connections with databases), Information Quality (through connections with live, operational feeds), Knowledge Quality (through modeling and simulation using predictive simulation, and Awareness Quality (through the use of its real-time simulation element as a state estimator). While the DSAP Framework matches up as a support tool for the GIG, support for a web services paradigm is needed. This support is discussed below.

In order to address these concepts, the DSAP Framework is being enhanced to work with Network Centric Enterprise Services (NCES) in a Service Oriented Architecture in order to support applications utilizing the Global Information Grid (GIG). This work includes (1) defining the granularity of services to which elements of the DSAP Framework will be broken down for use in a GIG environment; and, (2) defining the types and format of data and messages that will be transmitted between elements of the infrastructure and resources residing on the GIG. These are discussed in Sections 4.1 and 4.2.

4.1 Defining the Granularity of Services for the GIG

The first objective in making the DSAP Framework "GIG-ready" is to identify the granularity of the web services that will be provided. There are two ways that DSAP can be viewed in this process (1) viewing DSAP as a coarse-grained tool that will play, as a whole, in the GIG environment, or (2) a fine-grained set of services, each of which can play in the GIG environment.

4.1.1 Coarse-grained Support for DSAP

This effort focused on enabling coarse-grained support for DSAP in a GIG environment. In this manner, the DSAP Framework required use of services that enabled it to (1) initialize scenarios, (2) define plans and objectives, (3) prune unnecessary or failing plans when it interacts with the outside world, and (4) calibrate with updated C4I information. In addition, it also required the use of simulation services to fulfill the needs of its simulation components.

In terms of the services that DSAP provides to the outside world, the DSAP Framework can (1) execute simulations and replications and provide intermediate results, (2) evaluate simulations and replications and provide evaluation results, and (3) provide state estimation information.

4.1.2 Fine-Grained Support for DSAP

This effort outlined the implementation of a fine-grained solution for DSAP that will be implemented after the coarse-grained implementation. This finer-grained approach consists of making the individual elements of the MRF web-enabled. In this process, each Worker/Tasker/Manager in the MRF provides a simulation/replication execution or evaluation services. These services and the data and message structure they require can be defined presented to the GIG community to identify how potential DSAP users can interact with the framework across a distributed enterprise.

4.2 Defining the Data and Services for GIG Use

This effort defined the types of services that the DSAP Framework will support in a web environment. Subsequent steps for making DSAP GIG-enabled entail describing the structure of data and web services required to play in a GIG environment. There are four steps to this process:

1. Structuring and describing the information to be exchanged
2. Specifying the web service
3. Accessing and communicating with the web service
4. Registering and locating web services

Each of these is described in the following subsections.

4.2.1 Structuring Information to Be Exchanged

The first step in defining the necessary data structure and services to make DSAP "GIG-ready" involves structuring and describing the information that will be passed between elements of the DSAP system. This includes defining the structure and type of data necessary to initialize scenarios, describe plans, identify plan objectives and plan priorities, encapsulate plan and replication results, provide plan evaluation information, and extract real-time information between the different services provided by the DSAP Framework. A major emphasis in this area is utilizing a format that can (1) be understood by Commanders and their staff at Air Operations Centers, and (2) interact with tools utilized by Commanders and their staff at Air Operations Centers. The primary mechanism for describing information in DSAP is the Extensible Markup Language (XML). XML provides the description of data to be exchanged as well its storage and transmission formats. A key facet of XML is that it allows for the definition of schemes that can be used to define supported data types, content, and structure.

With this in mind, several languages and data models have been examined for use in supporting DSAP in a GIG environment. These include: Military Scenario Description Language (MSDL), Battle Management Language (BML), and the Command and Control Information Exchange Data Model (C2IEDM). Each of these models/languages is described below, while outlining their use in the DSAP Framework.

4.2.1.1 Military Scenario Description Language (MSDL)

Military Scenario Description Language (MSDL) is a language used to initialize and load scenarios in a simulation environment through the use of an XML based data interchange format. This format enables Command and Control (C2) planning applications to interchange the military portions of scenarios with Simulations and other applications. MSDL targets the initialization of simulations and C2 systems with initial state and planned actions.

For DSAP, MSDL is used to define the initial simulation scenarios and initial plans and alternatives that the simulation elements will execute. MSDL also is used to define these scenarios from real-time C4I intelligence data, or to pull representative scenarios from a scenario databases. Use of MSDL enables DSAP users to utilize any MSDL-based scenario, or provide scenario information in a standard format to other users on the GIG.

4.2.1.2 Battle Management Language (BML)

The Battle Management Language is a vocabulary or lexicon used by simulation users and developers to specify how to plan and automate military functions in support of Battle Management activities. BML provides a data format for describing military behavior that is derived from military doctrine. The resulting description is a standard that can be passed from human to machine, or machine to machine. BML is used to (1) describe Command and Control forces and equipment conducting military operations, and (2) provide for situational awareness and a shared common operational picture.

The DSAP Framework implementation sends plan information, along with objectives and priorities throughout the infrastructure using messages formatted as comma-delimited spreadsheets. In the initial implementation, these messages were very difficult to decipher and did not adhere to any standard lexicon, making their use by Commanders and their staff at Air Operations Centers very difficult. To-be implementations of DSAP will alleviate this problem by utilizing BML to address these deficiencies by specifying the plans (COAs) and alternatives that will be executed by both the predictive simulation component and the simulation-based state estimator. BML will also be used to specify plan objectives and priorities. These descriptions are derived from military doctrine and will have much greater use when supporting activities at Air Operations Centers.

4.2.1.3 Command and Control Information Exchange Data Model (C2IEDM)

The Command and Control Information Exchange Data Model (C2IEDM) is an information exchange and data management model developed by NATO to specify the structure of information passed between Command and Control Information Systems (C2IS). The C2IEDM preserves the meaning and relationships of information to be exchanged between C2IS at the Conceptual, Logical, and Physical levels.

The DSAP Framework will use the C2IEDM in conjunction with MSDL or BML to specify the underlying relationships that populate its entities and actions for its simulation scenarios. The C2IEDM can be used for data interchange at both initialization and in extracting and posting intermediate and final results.

4.2.2 Specifying the Web Services

The second element defined for DSAP involves specifying and describing the types of web services present in DSAP. The types of web services addressed include scenario generation, plan generation, simulation/replication execution, simulation/replication evaluation, simulation/replication pruning, simulation results publication, simulation evaluation publication, state estimation publication, and simulation calibration. These services are described using Web Service Description Language (WSDL). WSDL is a standard web service specification language defined by the World Side Web Consortium (W3C) that describes the web service's functionality, location, content, constraints and input/output data structure.

4.2.3 Accessing and Communication with the Web Service

The third element of web services defined for the DSAP Framework with respect to its role in a GIG environment is the methods that are used to access and communicate with its web services. In the case of DSAP, future versions of the architecture will utilize Simple Object Access Protocol (SOAP) as the basic protocol for its Object Request Broker technology. This will

provide a message framework for exchanging data [5] and defining mappings for basic DSAP functions that initialize scenarios or plans and post simulation and evaluation result information to the GIG environment.

4.2.4 Registering and Locating Web Services

The fourth element of web services that has been defined for DSAP Framework with respect to the GIG environment is the Registering and Locating of DSAP Web Services. To support the registration of DSAP Web Services, DSAP will use Universal Distribution Discovery and Interoperability (UDDI). Use of UDDI enables the discovery and use of DSAP Web Services by other users residing on the GIG.

5.0 Future Work

The DSAP Framework is being used to provide a prototype Predictive Analysis and Real-time State Estimation capability for plans derived from Air Tasking Orders (ATOs) and their alternatives while calibrating those plans with real-time C4I database inputs. The DSAP Framework utilizes JSAF as both its predictive (faster-than-real-time) and real-time simulation components, and pulls C4I information from Theater Battle Management Core System's (TBMCS) Air Operations Database (AODB) and Modernized Integrated Database (MIDB). Additional simulation environments such as Air Warfare Simulation (AWSIM) and Force Structure Simulation (FSS) are being investigated for use as the predictive simulation component. In addition, this effort integrated DSAP with the web-service implementation of TBMCS (T-Bone) where web-services are being used to extract information regarding Red Force facilities and Blue Force tracks. This extraction is the first step in DSAP's eventual support for the Global Information Grid (GIG) through the use of web-service concepts.

To continue to build on successes of our DSAP work, future efforts will to continue to build on MRF functionality and update the working prototype of the DSAP Framework in the Modeling and Simulation (M&S) Facility at AFRL Rome Research Site. This future work will integrate DSAP with other software modules and systems participating in the Joint Synthetic Battlespace for Research and Development and target DSAP for participation in a military exercise or battle experiment. Specifically, future efforts will target the following enhancements and tasks based on our ongoing work in this area.

5.1 Improved Calibration and C4I Data Update

Future efforts will build on the current C4I calibration capability used by DSAP that employs either basic publish/subscribe mechanisms or utilizes web-services to update the running simulations. The current implementation demonstrates a C4I update capability by connecting to TBMCS 1.1.4 (T-Bone) and extracting minimal track information through available web services. Only minimal track information populates the current TBMCS implementation. For these enhancement efforts, we will work to extract more detailed information from the Track Management Database (TMDB) and potentially other modules (such as the SAA) to provide better, more timely, updates to the DSAP software. Based on our current research, part of this process may entail work with AFRL to populate tracks in the unclassified TBMCS that are relevant to our test scenarios used by DSAP/JSAF or other simulation environments. Thus, future efforts will entail working with the pertinent organizations to obtain the required access need to populate the databases.

5.2 Integrate with Additional JSB-RD Components

One of our main goals for the DSAP Framework is to take advantage of additional software components as part of the larger JSB-RD picture. DSAP functionality provides mechanisms to simulate, calibrate, and evaluate tasks based on scenarios, mission plans and user-defined priorities and objectives. Other elements of the JSB-RD include Visualization Modules, Plan Generation Tools, Plan Optimization Tools, Situational Awareness and Analysis tools and databases. Future efforts will work to define the interfaces between our existing DSAP components to make use of these components as part of a larger integrated picture. The resultant environment will then allow for extracting ATO information, generating plans from that information (3rd party software), simulating plans (COAs) and alternatives. RAM Laboratories

will also work with both contractor and AFRL personnel generating the visualization tools and the user interfaces to provide a consistent format and API for specifying plan objectives and priorities that can be passed to DSAP effectiveness modules.

5.3 Pushing Data

This effort determined that there is a need for pushing ISR situational awareness data in order to populate TBMCS or TBONE databases. Future efforts will need to investigate, obtain the proper access, and implement the required mechanisms for pushing simulated or estimated situational awareness data to TBMCS, TBONE or other outside databases residing on the GIG.

5.4 Installation of an Operational Capability

This effort installed DSAP's predictive and real-time state-estimation capability at AFRL. Future efforts will continue to update DSAP and will work to integrate DSAP with the other JSB-RD software modules at AFRL. The results will be a leave-behind capability that is integrated with JSAF 2004, T-Bone and potentially other simulation components that can be utilized by engineers, analysts, and developers in the laboratory.

5.5 Exercise Participation

A major goal for future efforts will be to integrate DSAP with other JSB-RD components and utilize DSAP to provide C2 Situational Awareness as part of an exercise or Battle Experiment. RAM Laboratories will work with the AFRL customer to identify the appropriate exercise (possibly a JFCOM hosted exercise) and develop both the underlying simulations and database inputs for use in that exercise. This will include working with AFRL and their customer to identify the exercise scenarios, objectives, and key data sources and ISR elements that will be considered by DSAP. Future efforts will entail working with available scenario generation tools and plan generation tools to implement the simulation in JSAF or other embedded simulation environments. Future efforts will also entail selecting and subscribing to the specific TBMCS (TBONE) data and working with AFRL and additional contractor personnel to provide the visualization and user-selection of plans and the desired situational awareness criteria.

6.0 Bibliography

John R. Surdu. Connecting Simulation to the Mission Operational Environment. Ph.D. Thesis. Texas A&M University. 2000

Alex F. Sisti. “Dynamic Situation Assessment and Prediction (DSAP)” Proceedings of SPIE, Enabling Technologies for Simulation Science VII Vol.5091. 2003.

Dr. Paul Phister, Dr. Timothy Busch, and Igor Plonisch. “Joint Synthetic Battlespace: Cornerstone for Predictive Battlespace Awareness.”

Reaper Jerome, Trevisani Dawn, and Alex Sisti. “Real-Time Decision Support System (RTDSS)” Proceedings of the Western MultiConference. January, 2003.

McGraw Robert, Lammers Craig, and Steinman Jeff, 2004. “Software Framework in Support of Dynamic Situation Assessment and Predictive Capabilities for JSB-RD”. In proceedings of the SPIE - Enabling Technologies for Simulation Science VIII Conference.

McGraw Robert, Lammers Craig, and Trevisani Dawn, 2004. “Dynamic Situation Assessment and Predictive Capabilities in Support of Operations”. In proceedings of the Fall Simulation Interoperability Workshop, Orlando, FL. 2004.

McGraw Robert, Lammers Craig, Steinman Jeff, and Trevisani Dawn, 2005. “A DSAP Framework for the Global Information Grid's Modeling and Simulation Community of Interest”. In proceedings of the *Spring Simulation Interoperability Workshop,* San Diego, CA. 2005.

Lammers Craig, McGraw Robert, and Trevisani Dawn, 2005. “Applying a Multireplication Framework to Support Dynamic Situation Assessment and Predictive Capabilities”. In proceedings of the SPIE - Enabling Technologies for Simulation Science IX, Orlando, FL. 2005.

Effects Based Operations. Available: http://www.afrlhorizons.com/Briefs/June01/IF00015.html

Theater Battle Management Core Systems (TBMCS). Available http://jitc.fhu.disa.mil/tbmcs/tbmcs.htm.

Available http://www.mstp.quantico.usmc.mil/modssm2/InfoPapers/INFOPAPER%20JSAF.htm

Numrich, S.K., Hieb, M., and Tolk, A. “M&S in the GIG environment: An Expanded View of Distributing Simulation” Presented at the Interservice Industry Training Simulation Education Conference. Orlando, FL. 2004.

http://e-mapsys.com/C2IEDM-MIP_Overview_20Nov2003.pdf

]Steinman Jeff, 2002. “The Standard Simulation Architecture.” In proceedings of the 2002 SCS Summer Computer Simulation Conference.

Douglas Schmidt. ACE+TAO. Available http://www.cs.wustl.edu/~schmidt/.

Bailey Chris, McGraw Robert, Steinman Jeff, and Wong Jennifer, 2001. "SPEEDES: A Brief Overview" In Proceedings of SPIE, Enabling Technologies for Simulation Science V, Pages 190-201.

RAM Object Request Broker Programming Guide, Version 1.2, DRAFT.

7.0 Acronyms

ACE	Adaptive Communication Environment
AODB	Air Operations Data Base
AFRL	Air Force Research Laboratory
ATO	Air Tasking Order
BML	Battle Management Language
C2IEDM	Command and Control Information Exchange Data Model
C2IS	Command and Control Information Systems
C4I	Command, Control, Communications, Computers, and Intelligence
CCSE	Common Component Simulation Engine
COA	Course Of Action
CORBA	Common Object Request Broker Architecture
DSAP	Dynamic Situation Assessment and Predictive
FSS	Force Structure Simulation
FTRT	Faster Than Real Time
GCCS	Global Command and Control System
GIG	Global Information Grid
GUI	Graphical User Interface
IFSB	Information Systems Branch
IITSEC	Interservice Industry Training Simulation Education Conference
JDBC	Java Data Base Connectivity
JDK	Java Developer's Kit
JSAF	Joint Semi-Automated Forces
JTIDS	Joint Tactical Information Delivery System
JWARS	Joint WarGaming System
MIDB	Modernized Integrated Data Base
MRF	Multiple Replication Framework
MSDL	Military Scenario Description Language
NATO	North Atlantic Treaty Organization
NCES	Network Center Enterprise Services
NCOW	Network Centric Operations and Warfare
ODBC	Open Data Base Connectivity

ORB	Object Request Broker
OS	Operating System
POC	Point of Contact
RT	Real Time
RTP	Real Time Picture
SATCOM	Satellite Communications
SBIR	Small Business Innovative Research
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SPEEDES	Synchronous Parallel Environment for Emulation and Discrete Event Simulation
TAO	The ACE ORB
TBMCS	Theater Battle Management Core System
XML	Extensible Markup Language